Teresa M. Corbin (SBN 132360)
Denise M. De Mory (SBN 168076)
Jaclyn C. Fink (SBN 217913)
HOWREY LLP
525 Market Street, Suite 3600
San Francisco, California 94105
Telephone: (415) 848-4900
Facsimile: (415) 848-4999

Attorneys for Plaintiff SYNOPSYS and Defendants AEROFLEX INCORPORATED, AEROFLEX COLORADO SPRINGS, INC., AMI SEMICONDUCTOR, INC., MATROX ELECTRONIC SYSTEMS, LTD., MATROX GRAPHICS INC., MATROX INTERNATIONAL CORP., and MATROX TECH, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,

Plaintiff,

vs.

AEROFLEX INCORPORATED, AMI SEMICONDUCTOR, INC., MATROX ELECTRONIC SYSTEMS LTD., MATROX GRAPHICS INC., MATROX INTERNATIONAL CORP., MATROX TECH, INC., AND AEROFLEX COLORADO SPRINGS, INC.,

Defendants.

SYNOPSYS, INC.,

Plaintiff,

vs.

RICOH COMPANY, LTD.,

Defendant.

Case No. C03-4669 MJJ (EMC)

Case No. C03-2289 MJJ (EMC)

[CORRECTED] DECLARATION OF DENISE M. DE MORY IN SUPPORT OF MOTIONS FOR SUMMARY JUDGMENT

I, Denise M. De Mory, declare as follows:

1. I am a partner at the law firm of Howrey LLP, counsel for Aeroflex Incorporated, Aeroflex Colorado Springs, AMI Semiconductor, Inc., Matrox Electronic Systems, Ltd., Matrox Graphics Inc., Matrox International Corp., and Matrox Tech, Inc. and Synopsys, Inc. (collectively, the "Defendants") in this action. The following declaration is based on my personal knowledge. If called upon to testify, I could and would competently testify to the matters set forth below.

2. Attached as Exhibit 1 is a true and correct copy of United States Patent No. 4,922,432.

3. Attached as Exhibit 2 is a true and correct copy of Ricoh's Patent Final Contentions Pursuant to Patent L.R. 3-6 served March 24, 2006.

4. Attached as Exhibit 3 is a true and correct copy of Ricoh's Supplemental Patent Final Contentions Pursuant to Patent L.R. 3-6 served June 23, 2006.

5. Attached as Exhibit 4 is a true and correct copy of Exhibit 65 to Ricoh's Supplemental Patent Final Contentions Pursuant to Patent L.R. 3-6 served June 23, 2006. [FILED UNDER SEAL]

6. Attached as Exhibit 5 is a true and correct copy of Ricoh's Claim Construction Opening Brief served on August 27, 2004.

7. Attached as Exhibit 6 is a true and correct copy of Responsive Claim Construction Brief for U.S. Patent No. 4,922,432 (Re-Filed) served on September 14, 2004.

8. Attached as Exhibit 7 is a true and correct copy of Ricoh's Claim Construction Reply Brief served on September 20, 2004.

9. Attached as Exhibit 8 is a true and correct copy of April 7, 2005 Claim Construction Order.

10. Attached as Exhibit 9 are true and correct copies of Ricoh's Expert Report of Mario Papaefthymiou on Infringement by Aeroflex (Ex. 9A), AMI (Ex. 9B) and Matrox (Ex. 9C) served June 23, 2006. [FILED UNDER SEAL]

11. Attached as Exhibit 10 are true and correct copy of Expert Deposition of Mario Papaefthymiou dated August 11, 2006. [FILED UNDER SEAL]

12. Attached as Exhibit 11 are true and correct copies of Ricoh's Expert Report of Donald Soderman on Infringement by Aeroflex (Ex. 11A), AMI (Ex. 11B) and Matrox (Ex. 11C). [FILED UNDER SEAL]

13. Attached as Exhibit 12 are true and correct copy of Exhibit 1 to Ricoh's Expert Report of Donald Soderman on Infringement by Aeroflex, AMI and Matrox. [FILED UNDER SEAL]

14. Attached as Exhibit 13 are true and correct copy of Expert Deposition of Donald Soderman dated August 14 & 15, 2006. [FILED UNDER SEAL]

15. Attached as Exhibit 14 are true and correct copy of Ricoh's Supplemental Responses to ASIC Defendants' Requests for Admissions.

16. Attached as Exhibit 15 are true and correct copy of April 26, 1989 Amendment to Prosecution History for U.S. Patent No. 4,922,432.

17. Attached as Exhibit 16 are true and correct copy of November 24, 1989 Amendment to Prosecution History for U.S. Patent No. 4,922,432.

18. Attached as Exhibit 17 are true and correct copy of United States Patent No. 6,226,776.

19. Attached as Exhibit 18 are true and correct copy of Article titled "Implementing 'C' Designs in Hardware : A Full-Featured ANSI C to RTL Verilog Compiler in Action" by Donald Soderman.

20. Attached as Exhibit 19 are true and correct copy of Article titled "Implementing C Designs in Hardware: A Full-Featured ANSI C to RTL Verilog Compiler in Action" by Donald Soderman.

21. Attached as Exhibit 20 are true and correct copy of Article titled "Implementing C Alogrithms in Reconfigurable Hardware using C2Verilog" by Donald Soderman.

22. Attached as Exhibit 21 are true and correct copy of Book titled "Introduction to HDL-Based Design Using VHDL" by Steve Carlson bearing bates numbers 2SP 0768299-2SP 0768485.

23. Attached as Exhibit 22 are true and correct copy of United States Patent No. 4,703,435.

24. Attached as Exhibit 23 are true and correct copy of Deposition of Yoon-Pin Simon Foo dated May 31, 2006. [FILED UNDER SEAL]

25. Attached as Exhibit 24 are true and correct copy of master Thesis by Yoon-Pin Simon Foo titled “Managing VLSI D Design Data with A Relational Database System” bearing bates numbers FOO 000038-FOO 000156.

26. Attached as Exhibit 25 are true and correct copy of University of South Carolina Class Enrollment Sheet for Topics/Computer Engr bearing bates number SC 0003406.

27. Attached as Exhibit 26 are true and correct copy of Project Description for ECE890B Spring 1986 titled “A Knowledge-based Behavioral Description Translator” bearing bates number FOO 000189-FOO 000191 and Foo Deposition Exhibit No. 521.

28. Attached as Exhibit 27 are true and correct copy of Article titled “A Framework for Managing VLSI CAD Data” by Yoon-Pin Foo and H. Kobayashi bearing bates numbers KBSC000904-KBSC000913 and Foo Deposition Exhibit No. 509.

29. Attached as Exhibit 28 are true and correct copy of Article titled “A Knowledge Based System for VLSI Module Selection” by Yoon-Pin Foo and H. Kobayashi bearing bates numbers KBSC000904-KBSC000913 and Foo Deposition Exhibit No. 508.

30. Attached as Exhibit 29 are true and correct copy of Deposition Transcript of Hideaki Kobayashi dated May 25, 2006. [FILED UNDER SEAL]

31. Attached as Exhibit 30 are true and correct copy of document bearing bates numbers KBCS00009-KBSC00028. [FILED UNDER SEAL]

32. Attached as Exhibit 31 are true and correct copies of Translation of bates numbered page RCL0011957B. [FILED UNDER SEAL]

33. Attached as Exhibit 32 are true and correct copy of Translation of bates numbered page RCL0011958B. [FILED UNDER SEAL]

34. Attached as Exhibit 33 are true and correct copy of document bearing bates number RCL0011963. [FILED UNDER SEAL]

35. Attached as Exhibit 34 are true and correct copy of Exhibit A to the JCC Statement.

36. Attached as Exhibit 35 are true and correct copy of Master Thesis by Thaddeus J. Kowalski titled “The VLSI Design Automation Assistant: A Knowledge-Based Expert System”.

37. Attached as Exhibit 36 are true and correct copy of Article titled “The VLSI Design Automation Assistant: From Algorithms to Silicon.

38. Attached as Exhibit 37 are true and correct copy of Deposition Transcript of Thaddeus J. Kowalski dated May 23, 2006. [FILED UNDER SEAL]

39. Attached as Exhibit 38 are true and correct copy of Request for Reexamination of United States Patent No. 4,922,432 to the United State Patent and Trademark Office dated January 17, 2006.

40. Attached as Exhibit 39 are true and correct copy of the United State Patent and Trademark Office Order Granting Reexamination of United States Patent No. 4,922,432 dated February 24, 2006

41. Attached as Exhibit 40 are true and correct copy of States Patent No. 4,922,432 File History.

42. Attached as Exhibit 41 are true and correct copy of document bates numbered RCL000064.

43. Attached as Exhibit 42 are true and correct copy of document bates numbered RCL000186.

44. Attached as Exhibit 43 are true and correct copy of Article titled “KBSC: A Knowledge Based Approach to Automatic Logic Synthesis” by H. Kobayashi bearing bates numbers KBSC000859-KBSC000872.

45. Attached as Exhibit 44 are true and correct copy of document bates numbered KBSC000870-KBSC000872

46. Attached as Exhibit 45 are true and correct copy of 37 C.F.R. 1.56(a).

47. Attached as Exhibit 46 are true and correct copy of document bates numbered RCL000228.

48. Attached as Exhibit 47 are true and correct copy of document bates numbered RCL000188.

49. Attached as Exhibit 48 are true and correct copy of document bates numbered RCL000197.

50. Attached as Exhibit 49 are true and correct copy of 37 C.F.R. 1.97-1.99.

51. Attached as Exhibit 50 are true and correct copy of the corrected second supplemental Smith product declaration.

52. Attached as Exhibit 51 are true and correct copy of Deposition Transcript of Cliff Warren dated June 6, 2006. [FILED UNDER SEAL]

53. Attached as Exhibit 52 are true and correct copy of Deposition Transcript of David Tran, dated December 16, 2005 and June 6, 2006. [FILED UNDER SEAL]

54. Attached as Exhibit 53 are true and correct copy of Deposition Transcript of Reed Packer dated June 8, 2006. [FILED UNDER SEAL]

55. Attached as Exhibit 54 are true and correct copy of Expert Report of R. Fred Lipscomb. [FILED UNDER SEAL]

56. Attached as Exhibit 55 are true and correct copy of Expert Report of Maureen S. Loftus. [FILED UNDER SEAL]

57. Attached as Exhibit 56 are true and correct copy of Deposition Transcript of David Chiappini, Vol. 1 dated February 23, 2006. [FILED UNDER SEAL]

58. Attached as Exhibit 57 are true and correct copy of Deposition Transcript of David Chiappini, Vol. 3 dated June 7, 2006. [FILED UNDER SEAL]

59. Attached as Exhibit 58 are true and correct copy of Erik Olson Declaration in Support of Motion for Summary Judgment of Noninfringement under 35 U.S.C. Section 271(g).

60. Attached as Exhibit 59 are true and correct copy of Michael Heynes Declaration in Support of Motion for Summary Judgment of Noninfringement under 35 U.S.C. Section 271(g).

61 Attached as Exhibit 60 are true and correct copy of Second Supplemental Declaration of David Chiappini of Matrox Graphics, Inc.

62. Attached as Exhibit 61 are true and correct copy of Second Supplemental Declaration of Eric Boisvert of Matrox Electronic Systems.

63. Attached as Exhibit 62 are true and correct copy of document bearing bates numbers MGI0688426-MGI0688481. [FILED UNDER SEAL]

64. Attached as Exhibit 63 are true and correct copy of Schedule 2.1.1 the the Wagner Expert Report. [FILED UNDER SEAL]

65. Attached as Exhibit 64 are true and correct copy of Deposition Transcript of Eric Boivert dated June 6, 2006. [FILED UNDER SEAL]

66. Aeroflex performs ASIC design work for United States Government sub-contractors operating under a Government prime contract. Contracts executed for such work incorporate an authorization and consent clause (Federal Authorization Clause 52.227-1) either in the purchase order or contract between Aeroflex and the sub-contractor or in the U.S. Government prime contract under which the design work is performed. At least $4,804,851 of Aeroflex sales are pursuant to government contracts incorporating an express authorization and consent clause.

67. Aeroflex internally classifies its government contracts with a prefix of “gv” before the contract number. Some government prime contracts referenced on purchase orders for product sales pursuant to contracts designated “gv” for products at issue likely contain an authorization and consent clause. In some cases, the subcontractors are Sandia National Laboratory, Los Alamos National Laboratory, and Boeing Satellite Systems, for example. At least one prime contract was not locatable through a Freedom of Information Act request.

68. Attached as Exhibit 65 are true and correct copy of Complaint dated January 21, 2003.

69. Attached as Exhibit 66 are true and correct copy of document bearing bates numbers KBSC001109-KBSC001117.

70. Attached as Exhibit 67 are true and correct copy of translations of bates number 2SP0708285. [FILED UNDER SEAL]

71. Attached as Exhibit 68 are true and correct copy of Chart of Synopsys-Ricoh Contract.

72. Attached as Exhibit 69 are true and correct copy of document bearing bates numbers SP00001-00032. [FILED UNDER SEAL]

73. Attached as Exhibit 70 are true and correct copy of Verilog HDL Compiler Reference Manual bearing bates numbers SP04436-SP04536. [FILED UNDER SEAL]

74. Attached as Exhibit 71 are true and correct copy of Product of the Year Awards document bearing bates numbers SP04649-SP04651.

75. Attached as Exhibit 72 are true and correct copy of Synopsys ASIC Partnerships bearing bates numbers SP04707-SP04727. [FILED UNDER SEAL]

76. Attached as Exhibit 73 are true and correct copy of Article by Ann Steffora "Avant! Shakes Up Front-End Design" – Juniper CAE Software – Product Announcement, Electronic New, June 21, 1999

77. Attached as Exhibit 74 are true and correct copy of Article by Ray Weiss "Hot Design Comb", Electronic Engineering Times, May 14, 1990

78. Attached as Exhibit 75 are true and correct copy of Article "CAE Software Gould…, Electronic News, July 1, 1991".

79. Attached as Exhibit 76 are true and correct copy of document bearing bates numbers 2SP0763439-2SP763460.

80. Attached as Exhibit 77 are true and correct copy of AMIS website print out.

81. Attached as Exhibit 78 are true and correct copy of AMIS website print out.

82. Attached as Exhibit 79 are true and correct copy of UTMC website print out.

83. Attached as Exhibit 80 are true and correct copy of Synopsys web site print out.

84. Attached as Exhibit 81 are true and correct copy of Synopsys web site print out.

85. Attached as Exhibit 82 are true and correct copy of Ricoh's Amended Complaint

86. Attached as Exhibit 83 are true and correct copy of document bearing bates number MGI0033893-MGI0033908. [FILED UNDER SEAL]

87. Attached as Exhibit 84 are true and correct copy document bearing bates number RCL011421-RCL011422. [FILED UNDER SEAL]

88. Attached as Exhibit 85 are true and correct copy document bates numbered SP0168741-SP0168776. [FILED UNDER SEAL]

89. Attached as Exhibit 86 are true and correct copy document bates numbered SP0167847-SP0167881. [FILED UNDER SEAL]

90. Attached as Exhibit 87 are true and correct copy document bates numbered SP0121654-SP0121667. [FILED UNDER SEAL]

91. Attached as Exhibit 88 are true and correct copy document bates numbered MGI0001490. [FILED UNDER SEAL]

92. Attached as Exhibit 89 are true and correct copy Deposition Transcript of Shir-Shen Chang dated January 5, 2006. [FILED UNDER SEAL]

93. Attached as Exhibit 90 are true and correct copy Deposition Transcript of Karen L. Pieper dated December 12, 2005. [FILED UNDER SEAL]

94. Attached as Exhibit 91 are true and correct copies of excerpts from the document entitled “Ricoh’s Written Report of Donald Soderman in Rebuttal to Reports of Kowalski, Mitchell and Van Horn.”

Executed this 18th day of August, 2006, at San Francisco, California.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

/s/Denise M. De Mory
Denise M. De Mory

EXHIBIT 43

International Journal of Computer Aided VLSI Design 1. 377–390 (1989)

KBSC: A Knowledge-Based Approach to Automatic Logic Synthesis

HIDEAKI KOBAYASHI
International Chip Corporation

TERUMI SUEHIRO AND MASAHIRO SHINDO
Ricoh Company, Ltd.

A knowledge-based silicon compiler (KBSC) has been jointly developed by Ricoh Company and International Chip Corporation. KBSC provides designers with a front-end graphic interface to automatic logic synthesis. A rule-based expert system synthesizes both data-path and control circuits. KBSC automatically translates an algorithmic description in flowchart form into a logic circuit (netlist) that consists of previously registered cells. ASIC design by KBSC is compared with design by a senior engineer in terms of design time, chip performance, and gate count. An inference engine chip for real-time rule processing is used as a design example. KBSC is also compared with other logic synthesis systems.

ASIC automatic logic synthesis flowchart input form
rule based expert systems silicon compilation

1 INTRODUCTION

Cost per large-scale integrated (LSI) chip can be defined by development cost D, fabrication cost F, and total production volume N. Therefore, chip cost C can be estimated by $C = D/N + F(N)$, where F is the function of N. If LSI circuits are produced in large volume, the development cost per chip is negligible. LSI circuits of this type include memories and other standard products.

ASICs are produced in small volume to meet customers' needs. In the case of ASICs, however, it becomes important to reduce development cost or design time. The process of ASIC design is divided into functional, logic, and layout stages. To reduce time required for functional and logic design, it is important to

1. Provide an optimum input form to specify an initial chip function.
2. Verify an initial chip function to ensure functional correctness.
3. Automatically translate a verified chip function to a logic circuit or netlist.

Correspondence and requests for reprints should be sent to Hideaki Kobayashi, International Chip Corporation, AT & T Building, 1201 Main St., Suite 2000, Columbia, SC 29201.

Manuscript received November 15, 1988; Manuscript revised June 5, 1989.

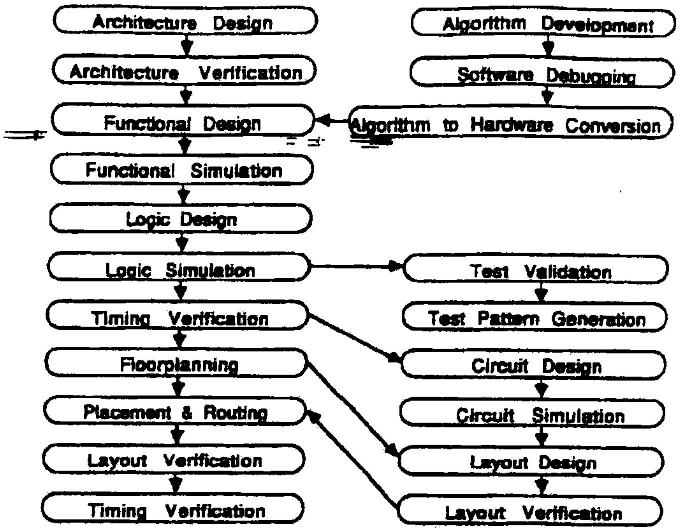


Figure 1. ASIC Development Flow.

Furthermore, to reduce time required for layout design, automatic translation of logic circuit (netlist) information to geometrical mask data is needed.

Many non-commercial logic synthesis systems [1–8] have been developed to translate automatically a behavioral or functional description into a logic circuit or chip layout. The concept of a knowledge-based silicon compiler (KBSC) was introduced in 1987 [9]. KBSC was developed jointly by Ricoh Company and International Chip Corporation (ICC). IF–THEN-type rules have been extracted from expert ASIC designers over several years at Ricoh and ICC, and stored in a knowledge base within KBSC. KBSC and other CAD tools are integrated into a comprehensive CAD system that provides designers with optimum input forms, such as flowcharts, state-transition equations, and functional module diagrams for cell-based design.

2 DESIGN METHODOLOGY

A typical ASIC development flow is shown in Figure 1. First, an entire system function is partitioned into a set of subfunctions (functional blocks). These functional blocks are then defined by various input forms, Figure 2.

1. Boolean expression or truth table form is for combinatorial circuit design.

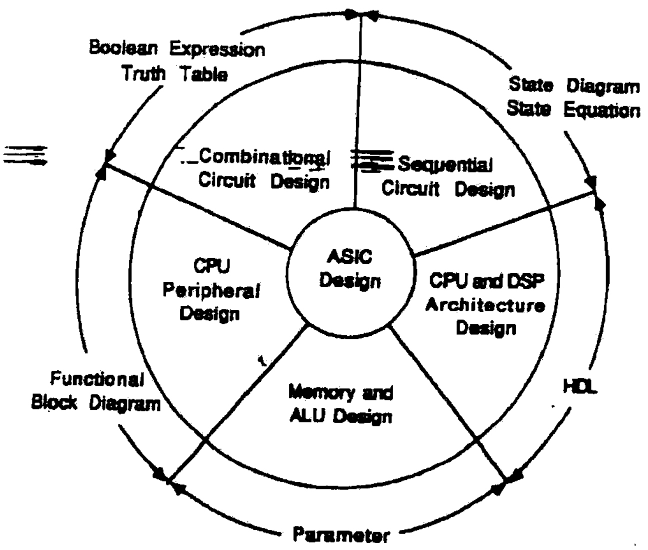


Figure 2. Input Forms for ASIC Design.

2. State-transition diagram or state-equation input form for sequential circuit design.
3. Parameter input form for memory and arithmetic and logic unit (ALU) design with variable address and data lengths.
4. Functional module input form using preregistered cells with bus compatibility for CPU peripheral LSI design.
5. Hardware description language (HDL) input form for CPU and digital signal processing (DSP) architecture design and performance evaluation.

Inputs 1 and 2 above require synthesis and optimization of logic. Logic synthesis and optimization are not required for input 3 since modules are generated by design rules and other constraints. Input 4 utilizes existing logic circuits corresponding to every module. Logic synthesis and optimization are not required since these circuits are already optimized. It is important to select an appropriate input form to design each functional block. The entire system design is completed by interconnecting these functional blocks.

KBSC's flowchart input form (Figure 4) is vital for "system" design that includes both data-path and control logic synthesis. KBSC flowcharts are useful for designers who are familiar with software programming as well as hardware system design. Flowchart-to-netlist conversion is achieved by an expert system where IF–THEN-type rules for logic synthesis are stored in a knowledge base. These rules are extracted from expert ASIC designers at Ricoh and ICC. KBSC's cell-based design approach provides quick turnaround time and design flexibility. Logic circuits designed by KBSC contain previously registered cells, each with information on its own mask data. Therefore, only placement and routing of these cells need to be performed by layout tools.

Layout design is performed in a hierarchical manner. First, highest level cells (functional blocks) are placed on a chip. An optimum placement is achieved by our floor planner, based on information such as each block area (transistor count), *X/Y* dimension ratio, and a netlist between blocks. The floor planner computes relative *X/Y* coordinates, routing areas between blocks, and information about terminal locations in each block.

After initial placement (floor planning), automatic routing takes place. Our automatic place–route software can handle macrocells with four-sided terminals as well as cells with only upper and lower terminals. Savings of 20% or more of the chip area are achieved by this approach compared with our conventional place–route software for two-sided terminals. To achieve 100% routability, routing area is estimated by heuristics. Extra routing area is eliminated by compaction. Layout of lower-level blocks is performed in a similar manner.

A data base stores cells with circuit information and physical mask data. Each cell contains information on logic symbols for schematic capture, models for logic stimulation, terminal names and locations for automatic placement and routing, cell sizes, and physical mask data. These cells are used for automatic logic synthesis as well as schematic capture to design logic circuits.

3 SYSTEM CONFIGURATION

The system configuration of KBSC is shown in Figure 3. The function of each software module in KBSC is explained.

The Flowchart Editor is an interactive functional editor for creation and modification of KBSC flowcharts for design specification entry. An example of a KBSC flowchart is shown in Figure 4. Actions and conditions as well as state transitions are represented by functional macros, which are independent of hardware. These macros are used to define data and numerical operators (e.g., add, subtract, shift, logical AND, logical OR) in each state. A sample list of macros is given in Figure 5.

The Flowchart Simulator is a verification tool for edited flowcharts at the functional level. It guarantees that edited flowcharts represent correct functions. ASIC users easily can operate the Flowchart Simulator to verify functional correctness.

After simulation, flowchart information is separated into actions and conditions. Actions are used for data-path synthesis, and conditions are used for control logic synthesis. Both actions and conditions are described

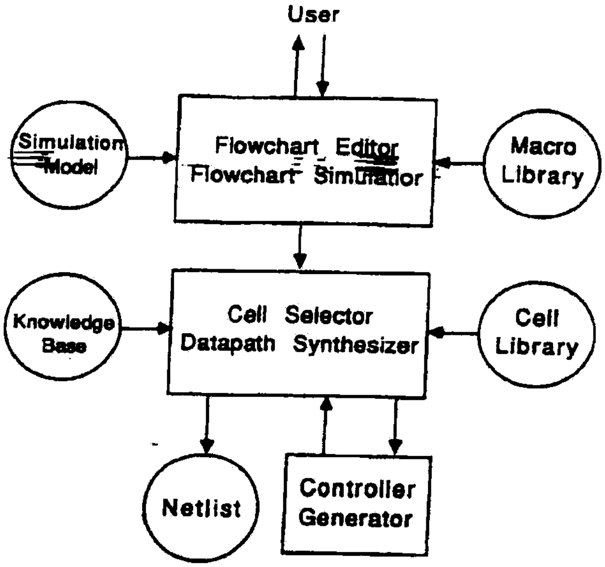


Figure 3. KBSC Configuration.

using the antecedent action form (AAF). An example of AAF is shown in Figure 6.

The Datapath Synthesizer provides automatic logic synthesis and optimization for data-path circuits. Synthesis and optimization of logic are accomplished by applying IF–THEN-type rules. Rules for placement and routing must be prepared for all macros. Example procedures for synthesis and optimization follow:

Place all macros according to rules for placement.
Place register blocks for all buses.
Route between all macros and register blocks. Add control signals to state-transition equations if needed.
Convert appropriate registers to counters and shifters. Add control signals to state-transition equations if needed.
Eliminate unnecessary macros and register blocks by classifying all macros and register blocks in terms of function and timing. Insert multiplexers into commonly used macros and register blocks, and add control signals to state-transition equations.
Connect clock signals. Clock signals for data-path circuits are synthesized separately from clock signals for control circuits. This is to assure that enable signals are stabilized before starting data transfer from data-path circuits.

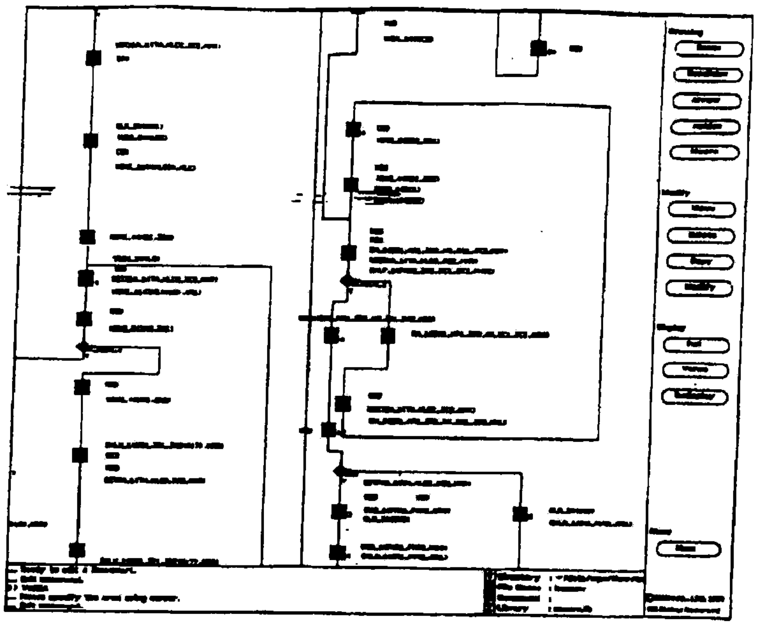
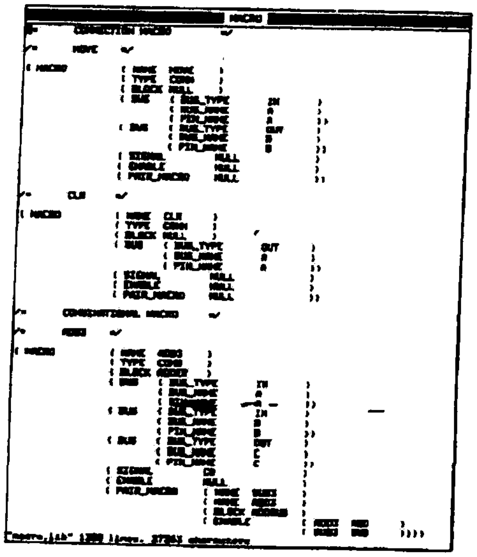


Figure 4. Example of a KBSC Flowchart.



```
                              MACRO
/*      CONNECTION MACRO          */
/*      MOVE      */
( MACRO          ( NAME   MOVE     )
                 ( TYPE   CONN     )
                 ( BLOCK  NULL     )
                 ( BUS    ( BUS_TYPE        IN       )
                          ( BUS_NAME        A        )
                          ( PIN_NAME        A        ))
                 ( BUS    ( BUS_TYPE        OUT      )
                          ( BUS_NAME        B        )
                          ( PIN_NAME        B        ))
                 ( SIGNAL           NULL             )
                 ( ENABLE           NULL             )
                 ( PAIR_MACRO       NULL             ))
/*      CLR       */
( MACRO          ( NAME   CLR      )
                 ( TYPE   CONN     )
                 ( BLOCK  NULL     )
                 ( BUS    ( BUS_TYPE        OUT      )
                          ( BUS_NAME        A        )
                          ( PIN_NAME        A        ))
                 ( SIGNAL           NULL             )
                 ( ENABLE           NULL             )
                 ( PAIR_MACRO       NULL             ))
/*      COMBINATIONAL MACRO       */
/*      ADDD      */
( MACRO          ( NAME   ADDD     )
                 ( TYPE   COMB     )
                 ( BLOCK  ADDER    )
                 ( BUS    ( BUS_TYPE        IN       )
                          ( BUS_NAME        A        )
                          ( PIN_NAME        A        ))
                 ( BUS    ( BUS_TYPE        IN       )
                          ( BUS_NAME        B        )
                          ( PIN_NAME        B        ))
                 ( BUS    ( BUS_TYPE        OUT      )
                          ( BUS_NAME        C        )
                          ( PIN_NAME        C        ))
                 ( SIGNAL           C0               )
                 ( ENABLE           NULL             )
                 ( PAIR_MACRO       ( NAME   SUBD    )
                                    ( NAME   ADDD    )
                                    ( BLOCK  ADDSUB  )
                                    ( ENABLE         ( ADDD   ADD    )
                                                     ( SUBD   SUB    ))))
"macro.lib" 1200 lines, 37263 characters
```

Figure 5. Sample List of Macros.

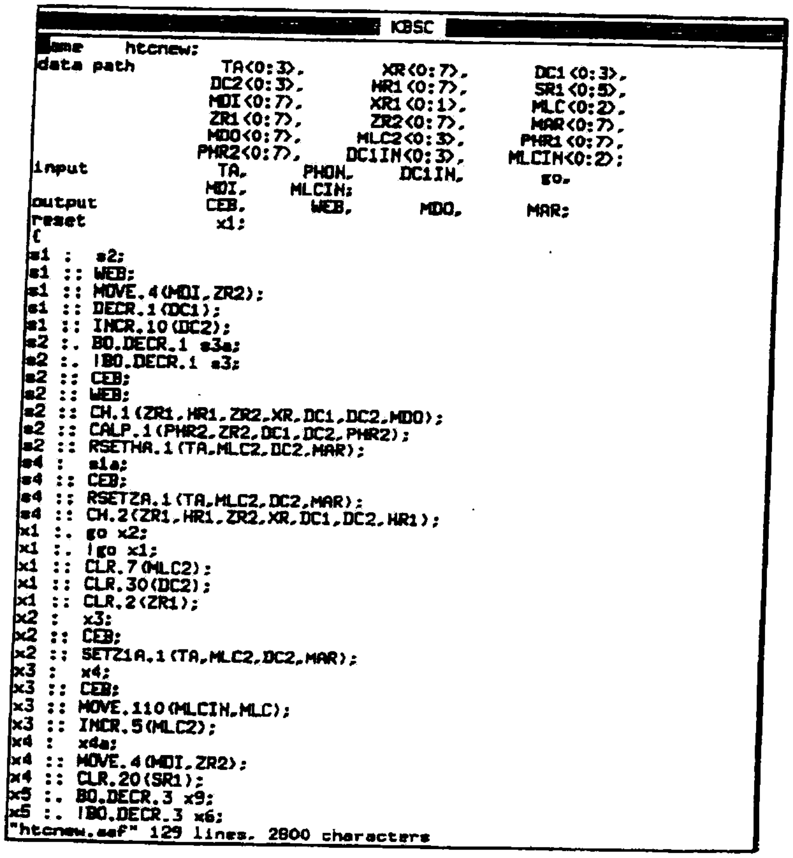
The Cell Selector uses rules to select existing cells from a library to replace functional cells (without geometrical information) used in data-path synthesis. Example procedures for cell selection follow:

Enter design constraints such as speed, power consumption, and chip area.

Select cells that satisfy entered design constraints.

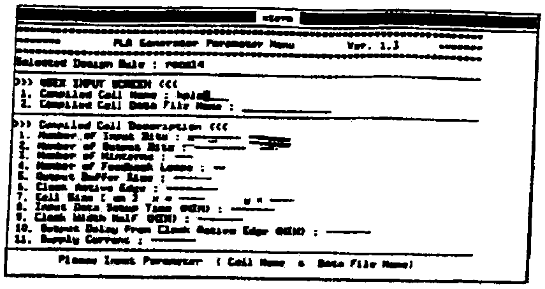
Eliminate unnecessary circuit portions if a selected cell has unused terminal(s).

The Controller Generator accepts state-transition equations (including modified state-transition equations during data-path synthesis) and auto-



```
                                   KBSC
name      htcnew;
data path              TA<0:3>,          XR<0:7>,         DC1<0:3>,
                       DC2<0:3>,         HR1<0:7>,        SR1<0:5>,
                       MDI<0:7>,         XR1<0:1>,        MLC<0:2>,
                       ZR1<0:7>,         ZR2<0:7>,        MAR<0:7>,
                       MDO<0:7>,         MLC2<0:3>,       PHR1<0:7>,
                       PHR2<0:7>,        DC1IN<0:3>,      MLCIN<0:2>;
input                  TA,       PHON,      DC1IN,           go,
                       MDI,     MLCIN;
output                 CEB,       WEB,        MDO,         MAR;
reset                  x1;
{
s1  :   s2;
s1  ::  WEB;
s1  ::  MOVE.4(MDI,ZR2);
s1  ::  DECR.1(DC1);
s1  ::  INCR.10(DC2);
s2  :.  BO.DECR.1 s3a;
s2  :.  !BO.DECR.1 s3;
s2  ::  CEB;
s2  ::  WEB;
s2  ::  CH.1(ZR1,HR1,ZR2,XR,DC1,DC2,MDO);
s2  ::  CALP.1(PHR2,ZR2,DC1,DC2,PHR2);
s2  ::  RSETHA.1(TA,MLC2,DC2,MAR);
s4  :   s1a;
s4  ::  CEB;
s4  ::  RSETZA.1(TA,MLC2,DC2,MAR);
s4  ::  CH.2(ZR1,HR1,ZR2,XR,DC1,DC2,HR1);
x1  :.  go x2;
x1  :.  !go x1;
x1  ::  CLR.7(MLC2);
x1  ::  CLR.30(DC2);
x1  ::  CLR.2(ZR1);
x2  :   x3;
x2  ::  CEB;
x2  ::  SETZ1A.1(TA,MLC2,DC2,MAR);
x3  :   x4;
x3  ::  CEB;
x3  ::  MOVE.110(MLCIN,MLC);
x3  ::  INCR.5(MLC2);
x4  :   x4a;
x4  ::  MOVE.4(MDI,ZR2);
x4  ::  CLR.20(SR1);
x5  :.  BO.DECR.3 x9;
x5  :.  !BO.DECR.3 x6;
"htcnew.aaf" 129 lines, 2800 characters
```

Figure 6. Example of Antecendent Action Form.



```
                              xterm
PLA Generator Parameter Menu                  Ver. 1.1
Selected Design Rule : [illegible]
>>> USER INPUT SCREEN <<<
 1. Compiled Cell Name : [illegible]
 2. Compiled Cell Data File Name : ________
>>> Compiled Cell Description <<<
 1. Number of Input Bits :
 2. Number of Output Bits :
 3. Number of Minterms :
 4. Number of Feedback Lines :
 5. Output Buffer Size :
 6. Clock Active Edge :
 7. Cell Size [ um ]  X = ____          Y = ____
 8. Input Data Setup Time (NS) :
 9. Clock Width Half (NS) :
10. Output Delay From Clock Active Edge (NS) :
11. Supply Current :
      Please Input Parameter ( Cell Name & Data File Name)
```

Figure 7. Parameter Menu for PLA Generation.

matically performs synthesis and optimization of logic circuits. Synthesis and optimization can be done in the following two stages.

1. Make a state assignment from state-transition equations and obtain Boolean expressions for circuits that implement state codes and output functions. Example procedures for making a state assignment follow:

   Count the total number of states and calculate the number of bits required to represent each state code.
   Assign a unique code for each state in order of appearance.
   Obtain Boolean expressions from a truth table describing present states, conditions, and next states.

2. Minimize Boolean expressions and generate a parameter menu for PLA generation (Figure 7), or generate a netlist for a logic circuit composed of existing cells. Example procedures for synthesizing a PLA-based system controller follow:

   Derive a truth table from Boolean expressions.
   Reduce the truth table by applying rules for data compression. Obtain a parameter file for PLA generation.

   Example procedures for system controller synthesis using random logic follow:

   Convert Boolean expressions to logic using only NAND gates.
   Minimize logic by applying rules for optimization.
   Convert logic to a circuit using NAND and/or NOR gates with less than or equal to six inputs each.
   Minimize logic by applying rules for optimization.
   Perform an error check for fan-out number excess and other checks.

A logic circuit (netlist) is generated automatically by combining optimized data-path and control circuits. An example of a logic circuit generated by KBSC is shown in Figure 8.

4 RULES FOR DATA-PATH SYNTHESIS

Example rule numbers used to synthesize data-paths are shown in Table 1 in execution (firing) order. The number of execution times for each rule is also shown in Table 2. Rule descriptions for data-path synthesis follow:

Rule 10 connects clock and reset terminals to data path circuits.
Rule 100 places an input buffer for an external input bus with a corresponding data length. It also places a register for an internal bus.
Rule 200 places macros (add, sub, and, or). If the macro's first operand is a bus, then it places a block with a corresponding data length. For example, add 0.12 (OP < 16:31 > C1) places a block with 16 bits of data length.
Rule 202 places macros (not, sfr, sfl). Similar to Rule 200.
Rule 220 places a macro (cmp). Similar to Rule 200.
Rule 230 places output buffers.

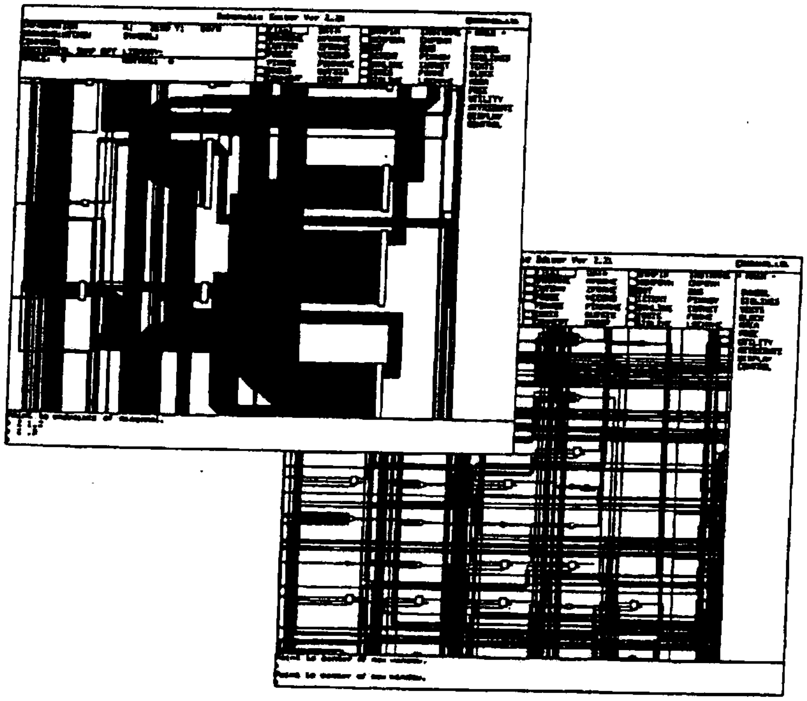


Figure 8. Example of a Logic Circuit Synthesized by KBSC.

Table 1. Rule Firing Sequence for Data-path Synthesis

CONTEXT	BEGIN	RULE	10
CONTEXT	BUSBLOCK	RULE	100
CONTEXT	MACROBLOCK	RULE	200
CONTEXT	MACROBLOCK	RULE	202
CONTEXT	MACROBLOCK	RULE	210
CONTEXT	MACROBLOCK	RULE	220
CONTEXT	MACROBLOCK	RULE	230
CONTEXT	MACROBLOCK	RULE	240
CONTEXT	CONNECTION	RULE	302
CONTEXT	CONNECTION	RULE	306
CONTEXT	CONNECTION	RULE	320
CONTEXT	CONNECTION	RULE	330
CONTEXT	CONNECTION	RULE	332
CONTEXT	CONNECTION	RULE	340
CONTEXT	CONNECTION	RULE	334
CONTEXT	CONNECTION	RULE	310
CONTEXT	CONNECTION	RULE	350
CONTEXT	CONNECTION	RULE	360
CONTEXT	CONNECTION	RULE	370
CONTEXT	CONNECTION	RULE	372
CONTEXT	CONNECTION	RULE	380
CONTEXT	TRANSFORM	RULE	400
CONTEXT	TRANSFORM	RULE	410
CONTEXT	ADDCLOCK	RULE	500
CONTEXT	ADDCLOCK	RULE	510
CONTEXT	ADDCLOCK	RULE	520
CONTEXT	SELECTION	RULE	610
CONTEXT	SELECTION	RULE	620
CONTEXT	SELECTION	RULE	630
CONTEXT	SELECTION	RULE	640
CONTEXT	SELECTION	RULE	650
CONTEXT	SELECTION	RULE	655
CONTEXT	SELECTION	RULE	660
CONTEXT	SELECTION	RULE	665
CONTEXT	SELECTION	RULE	670
CONTEXT	SELECTION	RULE	675
CONTEXT	SELECTION	RULE	680
CONTEXT	SELECTION	RULE	685
CONTEXT	SELECTION	RULE	642
CONTEXT	SELECTION	RULE	644
CONTEXT	SELECTION	RULE	690

Table 2. Number of Executions for Each Rule

RULE	10	TIME	1
RULE	100	TIME	1
RULE	200	TIME	2
RULE	202	TIME	3
RULE	210	TIME	1
RULE	220	TIME	1
RULE	230	TIME	4
RULE	240	TIME	1
RULE	[illegible]	TIME	1
RULE	300	TIME	1
RULE	302	TIME	1
RULE	306	TIME	3
RULE	310	TIME	1
RULE	320	TIME	1
RULE	330	TIME	9
RULE	332	TIME	4
RULE	334	TIME	1
RULE	340	TIME	1
RULE	350	TIME	4
RULE	360	TIME	1
RULE	370	TIME	3
RULE	372	TIME	7
RULE	380	TIME	10
RULE	400	TIME	2
RULE	410	TIME	1
RULE	500	TIME	2
RULE	510	TIME	1
RULE	520	TIME	8
RULE	600	TIME	1
RULE	610	TIME	8
RULE	620	TIME	5
RULE	630	TIME	2
RULE	640	TIME	1
RULE	642	TIME	3
RULE	644	TIME	4
RULE	650	TIME	1
RULE	655	TIME	1
RULE	660	TIME	3
RULE	665	TIME	1
RULE	670	TIME	3
RULE	675	TIME	1
RULE	680	TIME	3
RULE	685	TIME	1

Rule 240 places input buffers.
Rule 210 clears all flags to change mode from placement to routing.
Rule 300 routes macros (add, sub, and, or). It connects a bus register's output terminal, corresponding to the macro's first operand, to the macro's "A" terminal. Similarly, it applies to the macro's second and third operands.
Rule 302 is similar to Rule 300. It applies to the macro's second operand.

A total of 114 rules (42 different types of rules) are applied to synthesize data-paths automatically and to convert them to an equivalent circuit composed of existing cells.

5 RULES FOR CONTROL LOGIC SYNTHESIS

Example rule numbers used to synthesize control logic are shown in Table 3 in execution (firing) order. Rule descriptions for control logic synthesis follow:

Rule 10 constructs a truth table from Boolean expressions.
Rule 100 merges two rows in a truth table with the same combination of inputs into a single row.
Rule 500 checks the fan-out.

Table 3. Rule Firing Sequence for Control Logic Synthesis

CONTEXT	BEGIN	RULE	10	CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450
CONTEXT	BEGIN	RULE	20	CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	480
CONTEXT	MINIMIZE	RULE	100	CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450
CONTEXT	MINIMIZE	RULE	130	CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450
CONTEXT	MINIMIZE	RULE	100	CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450
CONTEXT	MINIMIZE	RULE	130	CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450
CONTEXT	MINIMIZE	RULE	130	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	OUT	RULE	210	CONTEXT	MDLMIN	RULE	400	CONTEXT	ERRCHK	RULE	500
CONTEXT	MINIMIZE	RULE	130	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	OUT	RULE	200	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	300	CONTEXT	MDLMIN	RULE	400	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	430	CONTEXT	MDLMIN	RULE	400	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	500
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	460	CONTEXT	ERRCHK	RULE	510
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	460	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	470	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	440	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450	CONTEXT	ADDBUF	RULE	600
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	480	CONTEXT	ADDBUF	RULE	610
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450				
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450				
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450				
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	480				
CONTEXT	MDLMIN	RULE	460	CONTEXT	MDLMIN	RULE	450				
CONTEXT	MDLMIN	RULE	400	CONTEXT	MDLMIN	RULE	450				

A total of 110 rules (18 different types of rules) are applied to synthesize control logic automatically and to convert it to a circuit using existing cells.

6 PERFORMANCE EVALUATION

ASIC design by KBSC was compared with design by a senior engineer in terms of design time, chip performance, and gate count. An inference engine chip for real-time rule processing is used as a design example.

1. *Design time:* KBSC needed only one-fifth of the design time required by a senior engineer. KBSC took approximately three weeks to complete the chip layout, whereas a senior engineer took approximately 15 weeks. Most of the time spent by the engineer was for architecture design.
2. *Chip performance:* The number of cycles to execute an IF–THEN-type rule is defined as a unit parameter. One rule was executed with 13 cycles for the circuit designed by KBSC, whereas the circuit designed by the senior engineer took 8 cycles to execute one rule. This was mainly due to the engineer's experience in designing circuits that can process more than two pieces of data in parallel.
3. *Gate count:* The KBSC circuit required 2,800 gates; the circuit designed by the engineer required 2,500 gates. The difference of 300 gates is due to the lack of rules for eliminating unused gates and optimizing logic circuits.

7 COMPARISON BETWEEN LOGIC SYNTHESIZERS

Logic synthesis systems in References [1–8] are compared with KBSC in terms of flowchart input form and rule-based approach to automatic data-path and control logic synthesis. A technology-oriented register transfer (RT)-level description is used in [1]. Input to KBSC is not an RT-level description, as used in many other approaches. KBSC's input is also not a flowchart like specification of control for typical finite-state machine design. Rules used in KBSC do synthesize both data-path and control circuits from a hardware-independent flowchart description. Output from KBSC is a netlist for automatic placement and routing.

Input to ALERT [2, 3] is an architectural description in Iverson's APL notation with declarations of variables to represent physical devices such as flip-flops and registers. The user of ALERT needs to specify memory size, word length, instruction format, and other hardware design features. In contrast, KBSC's input is a flowchart description with functional macros that are independent of hardware.

Input to the CMU-DA system [4] is an algorithmic description in ISP language. ISP description is translated to the first-level path graph with interconnections of abstract components. It is then mapped to the second level of the data-path graph with selected physical modules.

The Design Automation Assistant (DAA) [5] is an expert system that uses heuristic rules to synthesize architectural implementation from an algorithmic description with design constraints. Input to DAA is written in ISPS, a description for a Digital Equipment Corporation computer. The DAA produces a hardware network composed of modules, ports, links, and symbolic microcode.

Input to Flamel [6] is a behavioral description in the form of a Pascal program. Flamel's input is associated with a specified bus architecture. Functional blocks such as ALUs, registers, and I/O pads are ordered and placed on buses. Multiplexers are needed to regulate bus traffic at each block's input and output. In contrast, the KBSC approach is not tied to any fixed bus architecture and needs no multiplexers to regulate bus traffic.

SOCRATES [7] uses an expert system to optimize combinatorial logic for a specific target technology. Rules used in SOCRATES optimize gate-level circuits for speed and area in a given technology. A rule replaces a portion of a circuit by a functionally equivalent but more optimal circuit.

8 CONCLUSION

It has been shown that a knowledge-based silicon compiler (KBSC) dramatically reduces time required for functional and logic design. KBSC is clearly distinguished from other logic synthesis systems in terms of its flowchart input form and rule-based approach to automatic data-path and control logic synthesis. More than 10 chips have been designed using KBSC. Applications of these chips include speech synthesis, rule processing, vector-raster conversion, Japanese-character optical character recognition, and printer control.

From these initial results, we believe that KBSC is a useful tool for designers who are not familiar with hardware or circuit design. To improve the current version of KBSC more rules are needed for parallel processing and logic optimization. Most of the problems observed can be solved by simply adding new rules to the knowledge base and modifying rules. The program is written in C language and runs on Sun and other workstations supporting the UNIX operating system and the X windows system.

REFERENCES

[1] J.A. Darringer, and W.H. Joyner, Jr., "A New Look at Logic Synthesis," *Proc. of 17th Design Automation Conf.*, pp. 543–549, 1980.

[2] T.D. Friedman, and S.C. Yang, "Methods Used in an Automatic Logic Design Generator (ALERT)," *IEEE Transactions on Computers*, Vol. C-18, No. 7, pp. 593–614, July 1969.

[3] T.D. Friedman, and S.C. Yang, "Quality of Designs from an Automatic Logic Generator (ALERT)," *Proc. of 7th Design Automation Conf.*, pp. 71–89, 1970.

[4] A. Parker, D. Thomas, D. Siewiorek, M. Barbacci, L. Hafer, G. Leive, and J. Kim, "The CMU Design Automation System—An Example of Automated Data Path

Design," *Proc. of 16th Design Automation Conf.*, Las Vegas, NV, pp. 73–80, 1979.

[5] T.J. Kowalski, D.J. Geiger, W.H. Wolf, and W. Fichter, "The VLSI Design Automation Assistant: From Algorithms to Silicon," *IEEE Design and Test of Computers Magazine*, Vol. 2, No. 4, pp. 33–43, August 1986.

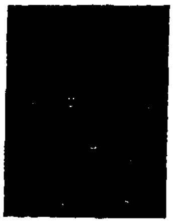
[6] H. Trickey, "Flamel: A High-Level Hardware Compiler," *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 2, pp. 259–269, March 1987.

[7] A.J. deGeus, and W. Cohen, "A Rule-Based System for Optimizing Combinational Logic," *IEEE Design and Test of Computers Magazine*, Vol. 2, No. 4, pp. 22–32, August 1986.

[8] L. Trevillyan, "An Overview of Logic Synthesis Systems," *Proc. of 24th ACM/IEEE Design Automation Conf.*, Paper 9.1, pp. 166–172, 1987.

[9] H. Kobayashi, "KBSC: A Knowledge Based Silicon Compiler—A Future Trend in ASIC Design," Ricoh ASIC Technical Seminar, Tokyo, Japan, October, 1987.

Hideaki Kobayashi (See guest editorial section (p. 355) of this special issue for author's biography).



Terumi Suehiro joined Ricoh Company in 1979 and has been involved in the development of CPU peripheral LSI. Since 1985 he has been developing a Ricoh CAD system for ASICs. He is currently Manager and Senior Engineer of the Semiconductors R & D Center. He received a B.S. degree in applied physics from Nagoya University in 1979.

Masahiro Shindo joined Ricoh Company in 1979 and has been involved in ASIC design, CAD tools, and strategic management. He is currently the Assistant General Manager of the Electronic Devices Division and Director of the Semiconductors R & D Center. Prior to joining Ricoh Company, he was employed by Mitsubishi Electric Corp. and was involved in the development of integrated circuit process technology (micron process, CVD) and devices (DRAM, SRAM, EPROM, LOGIC). He received a B.S. degree in industrial chemical engineering from the University of Ehime in 1983.

EXHIBIT
91

Gary M. Hoffman (*Pro Hac Vice*)
Kenneth W. Brothers (*Pro Hac Vice*)
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006
Telephone: (202) 785-9700
Facsimile: (202) 887-0689

Edward A. Meilman (*Pro Hac Vice*)
DICKSTEIN SHAPIRO LLP
1177 Avenue of the Americas
New York, New York 10036-2714
Telephone: (212) 835-1400
Facsimile: (212) 997-9880

Jeffrey B. Demain, State Bar No. 126715
Jonathan Weissglass, State Bar No. 185008
ALTSHULER, BERZON, NUSSBAUM, RUBIN & DEMAIN
177 Post Street, Suite 300
San Francisco, California 94108
Telephone: (415) 421-7151
Facsimile: (415) 362-8064

Attorneys for Ricoh Company, Ltd.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

RICOH COMPANY, LTD., Plaintiff, vs. AEROFLEX INCORPORATED, et al., Defendants	**CASE NO. C-03-4669-MJJ (EMC)** **CASE NO. C-03-2289-MJJ (EMC)**
SYNOPSYS, INC., Plaintiff, vs. RICOH COMPANY, LTD., Defendant	**RICOH'S WRITTEN REPORT OF DONALD SODERMAN IN REBUTTAL TO REPORTS OF KOWALSKI, MITCHELL AND VAN HORN**

I. INTRODUCTION

In addition to my testimony on infringement, I have been retained in this litigation by Ricoh Company, Ltd. ("Ricoh") to provide rebuttal testimony as to points raised in connection with the asserted invalidity and unenforceability of claims 13-17 of U.S. Patent No. 4,922,432 ("the '432 patent") in the written report of Thaddeus J. Kowalski (hereinafter referred to as "the Kowalski Report"), the written report of Tom M. Mitchell (hereinafter referred to as "the Mitchell Report"), and the written report of Charles E. Van Horn (hereinafter referred to as "the Van Horn Report").

As described below, in my opinion, nothing raised in either the Kowalski Report, Van Horn Report or the Mitchell Report clearly and convincingly challenges the validity or enforceability of any of claims 13-17 of the '432 patent.

II. COMPLETE STATEMENT OF OPINIONS

A. OVERVIEW OF STATE OF THE ART AT TIME OF THE INVENTION

The Mitchell Report asserts a misleading conclusion that "By December 1986, several systems including DDL/SX, SOCRATES, VEXED, DAS/Logic, and DAA were capable of synthesizing digital circuits from architecture-independent input specifications, using a rule-based expert system, and a predefined library of hardware cells, and had been described in publications in the open literature." I agree that, at this time there was a large community of Electronic Design Automation (EDA) developers and university researchers exploring various approaches to accelerate ASIC design, and were "capable" of immediately understanding the invention recited in claims 13-17 of the '432 patent. As will be evident from the detailed discussion in this Report, nothing in the prior attempts relied upon in the Mitchell Report boundaries of a single reference to provide missing disclosure of the claimed invention, the inquiry is not of "anticipation," but of "obviousness."

1. One of Ordinary Level of Skill in 1986

I do not agree with the Mitchell Report's contention that the typical level of ordinary skill in the art should be experts in the knowledge-based computer-aided software development and research, who were publishing papers at the Design Automation Conference or American Association for Artificial Intelligence annual meeting. The people conducting break-through research such as Drs. Kowalski and Mitchell were people of extraordinary skill in the art. Ordinary skilled people in the art at the time were merely programmers and engineers working on EDA (Electronic Design Automation) systems at the direction of highly-skilled designers. There was actually a broad base of EDA and ASIC design engineers working in this area. Many of these working engineers had BS degrees in Electrical Engineering or Computer Science and a couple of years of experience in the field (or MS degrees) as exemplified by several of the Synopsys Design Compiler developers Robert Walker, David Tran, Russ Segal, Janet Olson, Bharat Kalyanpur, David Clemans and Vasant Ramabadran.

2. VEXED

I do not agree with the Mitchell Report's contention that the "Vlsi Expert EDitor system" (hereinafter identified as the "VEXED system") described in the 1985 IEEE Transactions on Pattern Analysis and Machine Intelligence titled "A Knowledge-Based Approach to Design" (hereinafter "[Mitchell85]"), anticipates all the elements in claims 13, 15-17 of the '432 patent.[3] The VEXED system purports to be a knowledge based consultant system

[3] The Mitchell Report does not provide an opinion that claim 14 would be anticipated by this prior art. Accordingly, I have not been asked to provide any rebuttal opinions in this regard. Additionally, the Mitchell Report solely relies on [Mitchell85] in attempting to read the limitations of the pertinent claims of the '432 patent on the VEXED

that provides an interactive aid to the user in implementing a circuit given its function specifications. [Mitchell85] at 503. The circuit is initially represented as a black box. *Id.* The black box circuit is interactively replaced by the designer with submodules until the design is implemented with specific circuit elements (e.g., transistor networks, memories, etc.). *Id.* The VEXED system offers to the user different circuit implementations that may be used to provide the functionality required for the black box. The user selects one of the offered implementations, and the VEXED system outputs a schematic of circuit elements based on the implementations selected. [Mitchell85] at 506 and Fig. 4 (at 507).

The Mitchell Report fails to identify anything in the VEXED system that can be construed as "definitions of architecture independent actions and conditions," as required in claim 13 of the '432 patent. The CAM-CELL (Content Addressable Memory) example cited in the Mitchell Report is a special memory IC with trivial logic, not typical of ASIC designs that usually involve various logic configurations for control. The Mitchell Report tries to link the execution of the MEM-RULE and PASS-PAIR-RULE to the creation of two submodule abstractions and calls it a description of an architectural independent definition of an action and condition. The Mitchell Report makes reference to "actions and conditions" that are "described in the antecedents of these stored rules." Mitchell Report at 12. The fact, however, that any (alleged) "actions and conditions" are recited in a rule does not transform the actions and conditions into the required "definitions" of actions and conditions recited in claim 13. Thus, the

system. The Mitchell Report has briefly cited other references in the discussion of the VEXED system, specifically: "The CRITTER system – automating critiquing of digital circuit designs" Proceedings of the 21st IEEE Design Automation Conference, June, 1984; and "A Knowledge-Based Approach to VLSI CAD," in Proceedings of the 21st Design Automation Conference, IEEE and ACM, June 1984 (hereinafter respectively "[Kelly84]; and [Steinberg84]"). The Mitchell Report, however, does not expressly (or apparently) rely on any of these references in attempting to invalidate the claims of the '432 patent. For that reason, I have not commented on such references. I reserve the right to supplement this Report or otherwise give testimony in rebuttal to the extent any of those references or other evidence (e.g., prior art reference, testimony, etc.) is relied upon in attempts to show the presence (or obviousness) of claims 13-17 of the '432 patent based on the VEXED system.

Mitchell Report fails to cite anything in the VEXED system that would meet the claimed steps of "storing a set of definitions of architecture independent actions and conditions" and "specifying . . . one of said stored definitions."

Additionally, it would appear to one of ordinary skill in the art reviewing the disclosure of [Mitchell85] that the VEXED system operates at a technology-independent level. It does not synthesize "hardware cells" or produce a "netlist" of such "hardware cells," as required in the '432 patent. The term "hardware cells" as used in claim 13, for example, requires a representation of technology-specific, gate-level circuit elements that provide geometrical data (e.g., attributes such as width, height, technology, delay, power, etc.). *See, e.g.*, '432 patent, column 9, lines 35-51. The Mitchell Report cites no evidence that the VEXED system (as described in [Mitchell85]) stores such "hardware cells," uses "expert rules" to select such "hardware cells," or produces a "netlist" of such "hardware cells." There is no technology specific library of "hardware cells" composed of geometrical information of hardware components (e.g., NAND gates, NOR gates, etc.) from which to build an ASIC. The primitive pass transistors used in the VEXED system hardly qualify as a hardware cell since they have no specified area, performance, or power dissipation attributes. Any "rules" utilized by the VEXED system merely propose to the user one or more circuit implementations using generic (technology-independent) circuit elements. After user selection of the implementation (whether adopting a proposed rule or performing a manual implementation), the VEXED system represents as the final design, a circuit schematic showing technology-independent circuit elements (e.g., transistor networks, memory blocks, etc.). The Mitchell Report thus fails to cite anything in the VEXED system that meets the claimed steps of "storing . . . hardware cells," "storing . . . a set of rules for selecting hardware cells," and "selecting . . . a hardware cell" (which includes "generating . . . a netlist defining the hardware cells"), as recited in claim 13.

Claims 15-17 all depend (directly or indirectly) from independent claim 13, and include all of the limitations of claim 13. With respect to claims 15-17, therefore, the VEXED system lacks the same claim elements of claim 13, as noted above. The VEXED system, thus, cannot anticipate claims 15-17 for at least the reasons given above with respect to claim 13. In addition, it is noted that the Mitchell Report never attempts to map the limitations of claims 15-17 of the '432 patent to the VEXED system. Nothing in the VEXED system, thus, has been relied upon to meet the limitations of "generating data paths" (as in claim 15), "applying . . . a set of data path rules" (as in claim 16), or "generating control paths" (as in claim 17). It is therefore my opinion that the Mitchell Report fails to clearly and convincingly show that any of claims 15-17 are "anticipated" by the VEXED system.

For the reasons given above, it is my opinion that the Mitchell Report fails to clearly and convincingly show the VEXED system anticipates claims 13, 15-17 of the '432 patent.

3. DAS/Logic

I do not agree with the contention in the Mitchell Report that the system described in the conference paper titled, "DAS/Logic™: A Rule-based Logic Design Assistant," IEEE Second Conference on Artificial Intelligence Applications, December 11-13, 1985 (hereinafter "[Birmingham85]"), which is referred to in the Mitchell Report as the "DAS/Logic system," anticipates claims 13-17 of the '432 patent.[4] The DAS/Logic system is a tool for the design of integrated circuits from a high level logic description. [Birmingham85] at 264. The DAS/Logic system uses design knowledge encoded in expert rules to refine a textual behavioral description to a circuit schematic. *Id.* In particular, the DAS/Logic system accepts a behavioral description

[4] The Mitchell Report solely relies on [Birmingham85] in attempting to read the limitations of the pertinent claims of the '432 patent on the DAS/Logic system. I reserve the right to supplement this Report or otherwise give testimony in rebuttal to the extent other evidence (e.g., prior art reference, testimony, etc.) is relied upon in attempts to show the presence (or obviousness) of claims 13-17 of the '432 patent based on the DAS/Logic system.

Executed on July 24, 2006

Donald Soderman, PhD